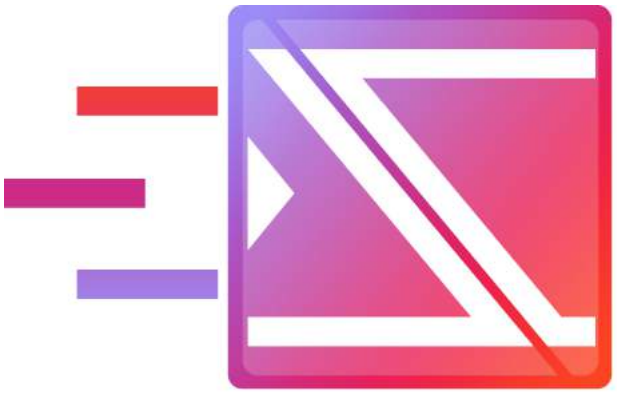


CIPHER

RAPPORT DE PROJET

DCipher project

Team Code Breakers

Dubiel Julien - Layet Nils - Florentin Dat - Jiang David

Rapport de projet

Table des matières

1 Introduction **5**

1.1 Qu'est-ce qu'un OCR ? . . . 5

1.2 Plan général de notre projet . . . 6

2 Présentation brève de l'équipe **7**

2.1 Code Breakers . . . 7

2.1.1 Dubiel Julien . . . 7

2.1.2 Layet Nils . . . 8

2.1.3 Florentin Dat . . . 8

2.1.4 Jiang David . . . 8

3 Pré-traitement de l'image **9**

3.1 Chargement de l'image . . . 9

3.2 Rotation . . . 10

3.2.1 Détection de l'angle d'inclinaison . . . 10

3.2.2 Rotation de l'image . . . 13

3.3 Binarisation . . . 14

3.3.1 La méthode d'Otsu . . . 15

3.4 Filtres . . . 17

3.4.1 Qu'est ce qu'un filtre ? . . . 17

3.4.2 Filtre linéaire . . . 17

3.4.3 Filtre médian . . . 18

3.4.4 Les Matrices de convolution . . . 19

4 Extraction des caractères **21**

4.1 Détection des blocs . . . 21

4.2 Détection des lignes . . . 22

4.3 Détection des caractères . . . 23

5 Reconnaissance des caractères **24**
5.1 Réseau de neurones . . . 24
5.1.1 Qu'est ce qu'un réseau de neurones ? . . . 24
5.1.2 Fonctionnement . . . 25
5.2 Présentation de l'algorithme . . . 25
5.3 Donnée centrale . . . 26
5.4 Principe d'apprentissage . . . 26
5.5 Principe de reconnaissance de caractère . . . 27
5.6 Inconvénients de cette méthode . . . 27
5.7 Perceptron . . . 28
5.8 Algorithme d'apprentissage . . . 32

6 Interface Graphique **33**
6.1 Description des composants . . . 33
6.2 Utilisation . . . 34
6.3 Gtk+ . . . 35
6.4 Correcteur orthographique . . . 36
6.4.1 Principe . . . 36
6.4.2 Distance de Levenshtein . . . 36

7 Problèmes rencontrés **38**

8 Site web **39**

9 Organisation **41**
9.1 Répartition des tâches . . . 41
9.2 Avancement de notre projet . . . 42

10 Dossier d'exploitation **43**
10.1 Manuel d'installation . . . 43
10.2 Manuel d'utilisation . . . 43

11 Impressions personnelles **45**
11.0.1 Dubiel Julien . . . 45
11.0.2 Layet Nils . . . 45
11.0.3 Florentin Dat . . . 46
11.0.4 Jiang David . . . 46

12 Conclusion **47**

13 Annexes **48**

Table des figures

1.1 Plan général de notre projet 6

3.1 Détection de blocs 10
3.2 Transposée de Hough 12
3.3 Rotation de l'image 13
3.4 Greyscale 14
3.5 Binarisation de l'image 16
3.6 Filtre Médian 18
3.7 Matrices de convolutions 19
3.8 Filtre Gaussien 20

4.1 Détection des lignes 22

5.1 Perceptron multicouches 28
5.2 Fonction sigmoïde et fonction de seuil 30
5.3 Schéma d'un perceptron monocouche 31

6.1 Interface graphique 34
6.2 Distance de Levenshtein 37

8.1 Version 1.0 du site 39
8.2 Version 2.0 du site 40

10.1 Ecran d'accueil de l'interface graphique 44

13.1 Interface graphique après avoir exécuté la binarisation 48
13.2 Après utilisation de détection des caractères 49
13.3 Ancien logo de notre projet 49

13.4 Ancienne bannière de notre projet 49
13.5 Nouveau logo de notre projet 50

Chapitre 1

Introduction

Ce rapport contient une vue d'ensemble du travail que nous avons réalisé pour notre projet d'info SPE : un OCR. Celui-ci englobera l'avancement de notre projet, la répartition des tâches, la façon dont les objectifs se remplissent, les difficultés rencontrées et comment celles-ci ont été surmontées. Nous avons tous plus ou moins touché à Ocaml et contrairement à l'année dernière, cette fois-ci le sujet n'est pas libre.

1.1 Qu'est-ce qu'un OCR ?

Le mot OCR (en anglais : Optical Character Recognition) signifie reconnaissance optique de caractères ou reconnaissance de texte, une technologie qui vous permet de convertir différents types de documents tels que les documents papiers scannés, les fichiers PDF ou les photos numériques, vers des formats modifiables et exploitables. Pour notre projet, nous devons faire un logiciel OCR capable de reconnaître du texte dans une image et de le convertir dans un fichier texte.

1.2 Plan général de notre projet

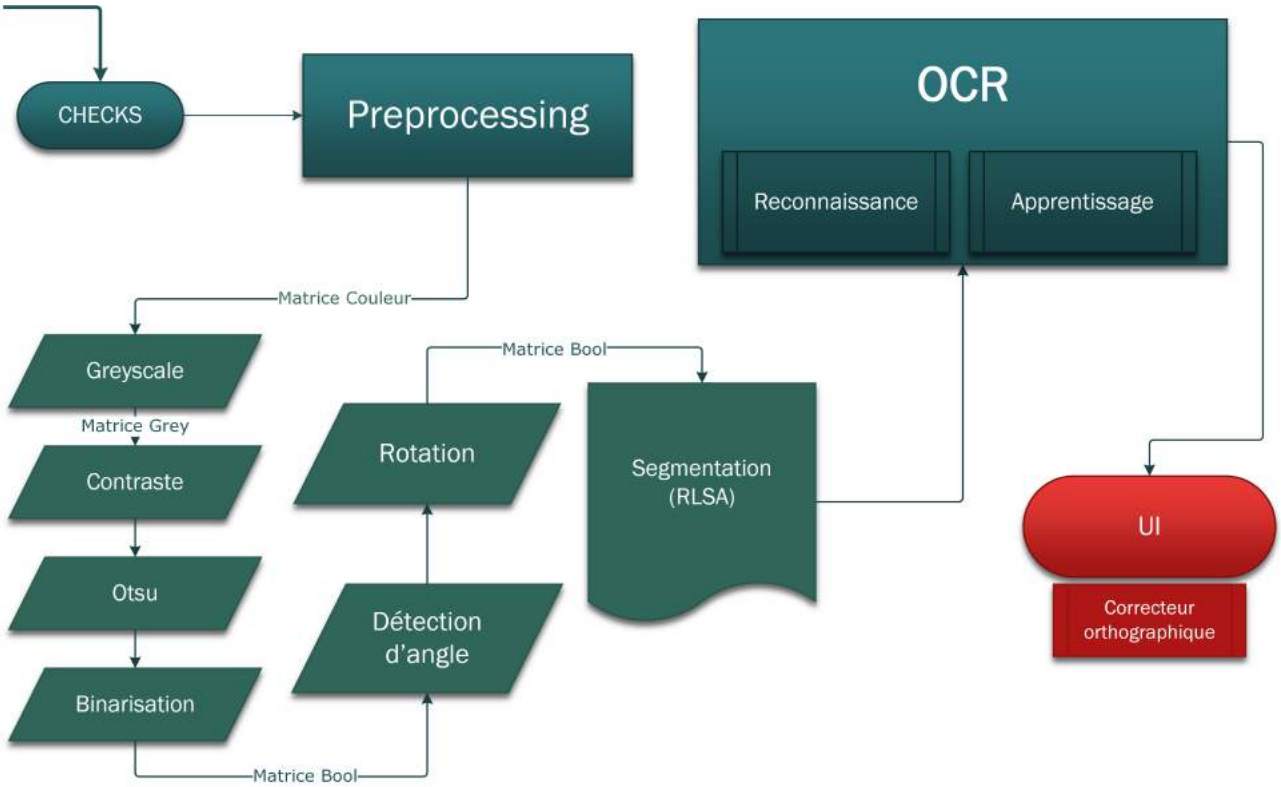


FIGURE 1.1 – Plan général de notre projet

Plan montrant les différentes étapes de notre OCR

Chapitre 2

Présentation brève de l'équipe

Dans cette partie se trouve la description de chaque membre de notre équipe, celle-ci est également disponible sur notre site internet décrit plus loin.

2.1 Code Breakers

Notre groupe est composée de 4 éléves d'info SPE dont 2 chinois. Le groupe s'est formé assez tôt dans l'année et nous nous sommes vite adapté au fait que la soutenance arrivait bientôt. Nous nous entendons bien dans l'équipe et faisons de notre maximum pour faire de ce projet une réussite. Pour une brève description de nos membres. Veuillez aller sur notre site internet dont le lien se situe dans la partie *"Site Web"*.

Notre chef de groupe et également créateur d'Epiportal par la même occasion est Julien Dubiel.

2.1.1 Dubiel Julien

Après une année à la tête d'Epiportal, il me fallait bien du nouveau dans ma vie de développeur ! Ce projet d'OCR en est l'occasion ! Bon, peut-être pas rêvée, mais certes, intéressante pour un projet de SPE, qui de plus, est pertinent relativement à l'apprentissage de ce langage repoussant qu'est le Caml :D. Je suis le chef de projet et je ferai donc le nécessaire pour garder une bonne cohésion de groupe, une bonne répartition des tâches et une excellente ambiance, afin d'avancer ensemble dans ce projet qui n'est pas des plus simples !

2.1.2 Layet Nils

Depuis que j'ai appris à développer en C/C++ durant une année passée aux Etats-Unis, j'ai été mordu par le code et l'informatique en général. Avec les cours donnés à EPITA, j'ai pu continuer à étendre (doucement) mes connaissances, et notamment par l'intermédiare de DCipher. Mon rôle dans le proet est avant tout de travailler sur le coeur de l'OCR (à savoir, ce que l'on nomme le fameux «réseau de neurones»). C'est de loin la partie que je trouve la plus intéressante, mais aussi sûrement la plus difficile. Je suis sûr que Dubiel me filera un coup de main de temps en temps :p

2.1.3 Florentin Dat

Après un an de ~~souffrance~~ à l'EPITA en tant que développeur d'un jeu vidéo, je suis de retour avec comme nouveau projet un OCR. Celui-ci doit être développé en OCaml... :vomi : Ce projet semble bien compliqué et un peu moins amusant qu'un jeu vidéo car nous sommes limités à des objectifs précis et nous faisons appel à moins de créativité ! J'espère que nous arriverons à nos fins ! C'est parti pour 3 mois de travail !

2.1.4 Jiang David

Et me voilà dans une nouvelle année à EPITA. Contrairement à ce que je pensais, ce projet est beaucoup plus intéressant qu'il n'en a l'air. L'avantage, c'est qu'il nous fait progresser en Caml et nous fait également bien réfléchir. Certes, nous devons nous plier aux conditions mais je pense que l'on devrait tous s'en sortir en travaillant ensemble.

Chapitre 3

Pré-traitement de l'image

Pour la partie pré-traitement de l'image, à partir d'un document A4 numérisé, plusieurs étapes sont nécéssaires. il faut tout d'abord griser l'image, en calculant pour chaque pixel sa valeur en dégradé de gris, puis il faut "nettoyer" l'image : il s'agit de gommer un maximum le bruit de fond, c'est-à-dire faire ressortir le tracé des caractères tout en supprimant le superflu autour pour donner un fond blanc en contraste par rapport aux caractères noirs. Par la suite, afin de soulager la mémoire et rendre les tâches ultérieures moins longues, nous binarisons l'image, c'est à dire que pour chaque pixel, sa valeur sera soit noire, soit blanche. Pour cette étape il est nécessaire de détecter à partir de quel niveau de gris chaque pixel sera noir, ou blanc. Après binarisation pixel par pixel, nous perdons souvent en précision du texte, c'est pourquoi nous appliquerons differents filtres si necessaire. Et enfin, une fois tout ceci fait, extraire l'image modifiée au format bitmap.

3.1 Chargement de l'image

Nous avions deux grands choix de bibliothèques pour le traitement de l'image : OCAMLSdl ou CamlImages.

Notre choix s'est porté sur **OCamlSDL** car celle-ci nous paraît à la fois puissante, rapide, et tout a fait adapté au traitement d'image. De plus c'est une bibliothèque entièrement libre et multi-plateforme.

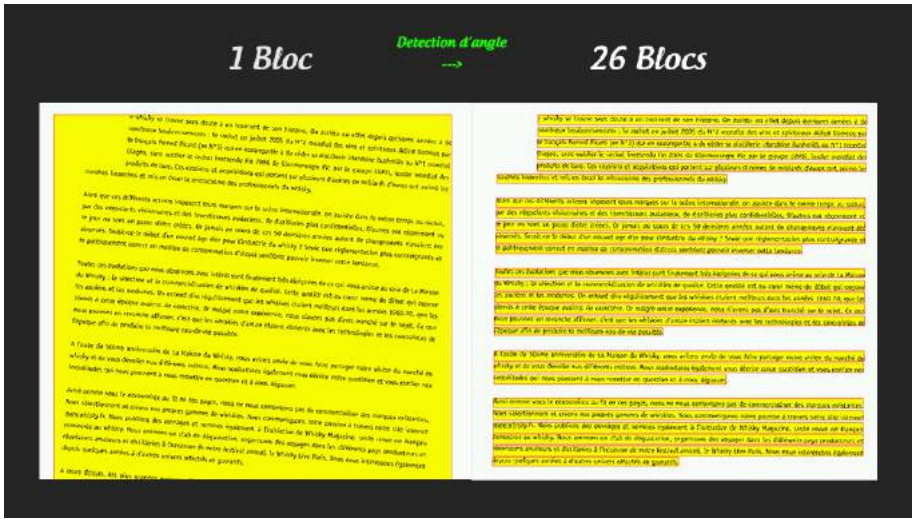


FIGURE 3.1 – Détection de blocs

3.2 Rotation

3.2.1 Détection de l'angle d'inclinaison

Pour détecter l'angle de rotation, nous étions tout d'abord partis sur une algorithme plutôt compliqué, parcourant les diagonales de l'images etc... Seulement celui-ci était dur à implémenter, long, pas très précis, voire aléatoire ! Nous avons aussi voulu implémenter la méthode de Hough mais vu les formules rencontrées sur internet, et le fait qu'elle ne soit pas des plus performantes, nous avons laissé la rotation de coté pour la première soutenance....

Et ce jusqu'à ce que nous réalisions la détection des lignes ! (cf "Détection des lignes et paragraphes") En effet, une idée très simple nous est venue à l'esprit : faire tourner l'image sur 180° en incrémentant l'angle de 1, 0.5, ou 0.25° à chaque fois, puis lancer la detection des lignes sur chaque rotation de l'image. Le résultat est tel que si l'algorithme de detection des lignes trouve un grand nombre de "bloc" (correspondant en fait aux lignes) alors c'est que l'image est bien orientée. Dans le cas où l'algorithme ne trouve qu'un seul et même bloc il y a beaucoup de chances pour que ce ne soit pas l'angle.

Cependant des inconvénients sont à citer ici : le fait de faire tourner (180 * précision) fois la matrice correspondant à l'image prend un temps considérable, ce n'est pas le cas de l'algorithme de détection des lignes qui lui est presque instantané.

Afin de résoudre ce problème, nous pensons, pour la seconde soutenance, parcourir non pas degré par degré, mais par dichotomie : c'est à dire que nous tournerons par exemple l'image de 30 degrés 6 fois puis les deux positons qui auront le plus de blocs indiqueront que c'est entre ces deux angles que se trouve l'angle parfait. Et ce itérativement jusqu'à trouver un angle présentant un maximum de blocs.

De plus en utilisant cette technique, la précision de l'angle trouvé serait beaucoup plus poussée. Un autre problème s'impose : le fait que l'algorithme détecte le texte quand il est droit, mais pas quand il est tourné à 180 degrés, c'est à dire complètement à l'envers. Afin de remédier à ce problème nous pensons que nous passerons d'abord par la detection des caractères qui dira oui ou non selon la probabilité que ce soit une lettre existante ou pas.

Pour notre deuxième et dernière soutenance, nous nous sommes enfin décidés à passer par la transformée de Hough.

La transformée de Hough inventée par Paul Hough en 1962 permet de détecter des droites, des cercles, des ellipses ou des objets plus complexes. Elle ignore assez bien le bruit et arrive a donner de bons résultats même si il manque quelques éléments dans une structure.

Son principe est le suivant : On suppose qu'il existe un nombre infini de lignes $y = ax + b$ (ou $r = xcos\theta + ysin\theta$ en coordonnées polaires) qui passent par un point mais que seul leur orientation (angle) diffère. Et si l'on trace r en fonction de θ on obtient une sinusoïde représentant l'ensemble des droites de paramètres(r, θ) passant par ce point. L'endroit ou ces droites se coupent est ce que l'on appelle "Espace de Hough".

Si l'on trace alors une autre sinusoïde pour un autre point, l'intersection des deux sinusoïdes donne la valeur de (r, θ) de la droite passant par ces deux points.

On va donc chercher à tracer pour l'ensemble des points chacune des sinusoïdes correspondantes dans l'espace de Hough. On cherchera alors à savoir quels sont les points ou les courbes se croisent pour ainsi en déduire la présence des droites.

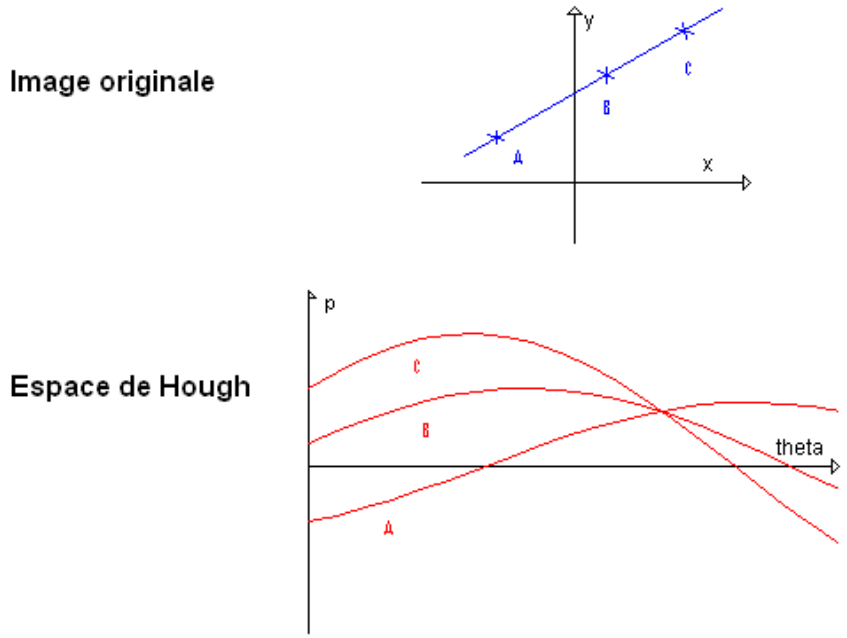


Schéma explicatif de la transposée de Hough

FIGURE 3.2 – Transposée de Hough

3.2.2 Rotation de l'image

La rotation de la matrice représentant l'image est tout simplement l'application d'une formule mathématique vue en cours de mathématique de SUP (comme quoi ça a du bon de suivre les cours de maths ! Hehe !). Il s'agit de se placer au centre de l'image (point de rotation) et pour chaque point de l'image de départ d'appliquer la formule suivante :

Où θ représente l'angle de rotation.

En application algorithmique, cela se traduit par :

$X = centreX + (pointX - centreX) * cosa - (pointY - centreY) * sina)$

$Y = centreY + (pointX - centerX) * sina + (pointY - centerY) * cosa)$

Puis nous détectons si le nouveau point alors à la position (X, Y) est bien dans la matrice de départ, et nous le plaçons dans une nouvelle matrice d'arrivée.

Mais il y a malheureusement des inconvénients à cette méthode.

Etant donné que nous partons de l'image de départ pour calculer les nouveaux points correspondants après rotation, nous perdons certains points et donc une certaine précision pour la détection des caractères. Afin de résoudre ce problème, pour la deuxième soutenance, nous allons calculer pour chaque point de l'image d'arrivée le point correspondant si nous avions fait une rotation classique.

Pour notre deuxième soutenance, nous avons décidé de passer par la rotation via la transformée de Hough qui a été expliquée un peu plus tôt.

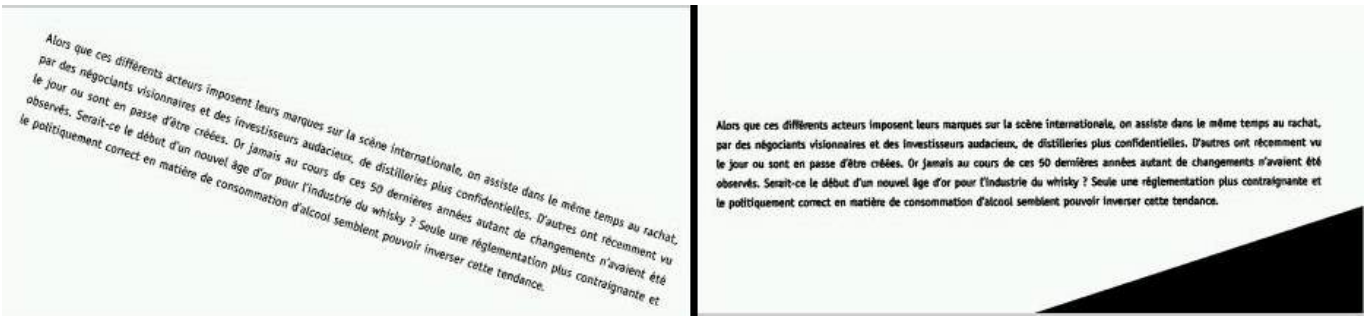

Alors que ces différents acteurs imposent leurs marques sur la scène internationale, on assiste dans le même temps au rachat, par des négociants visionnaires et des investisseurs audacieux, de distilleries plus confidentielles. D'autres ont récemment vu le jour ou sont en passe d'être créées. Or jamais au cours de ces 50 dernières années autant de changements n'avaient été observés. Serait-ce le début d'un nouvel âge d'or pour l'industrie du whisky ? Seule une réglementation plus contraignante et le politiquement correct en matière de consommation d'alcool semblent pouvoir inverser cette tendance.
Alors que ces différents acteurs imposent leurs marques sur la scène internationale, on assiste dans le même temps au rachat, par des négociants visionnaires et des investisseurs audacieux, de distilleries plus confidentielles. D'autres ont récemment vu le jour ou sont en passe d'être créées. Or jamais au cours de ces 50 dernières années autant de changements n'avaient été observés. Serait-ce le début d'un nouvel âge d'or pour l'industrie du whisky ? Seule une réglementation plus contraignante et le politiquement correct en matière de consommation d'alcool semblent pouvoir inverser cette tendance.

Image originale à gauche et après rotation à droite

FIGURE 3.3 – Rotation de l'image

3.3 Binarisation

Une image est représentée sous la forme suivante : elle est décomposée en un tableau de pixels, et chacun de ces pixels est généralement représenté par un mélange de trois composantes de couleurs qui sont le rouge, le vert et le bleu. Chaque composante a une valeur allant de 0 à 255 soit 256 valeurs différentes. Pour passer une image en niveau de gris, il suffit simplement d'effectuer un calcul très simple sur chaque pixel de l'image : (0.3 * (r) +. 0.59 * (g) + 0.11 * (b)) /. 255 Cette formule est trouvable partout sur internet, c'est un grand classique pour passer une image en niveau de gris.

Originale
Passage en greyscale
--->
Niveau de Gris
SECTION I AM
Associations
SOUS-PREFECTURE DE CARPENTRAS
Récépissé de Déclaration de CREATION
de l'Association N° 0843004852
Vu la loi du 1er juillet 1901 relative au contrat d'association ;
Vu le décret du 16 août 1901, portant règlement d'administration publique pour l'exécution de la loi précitée ;
Le Sous- Préfet de CARPENTRAS
donne récépissé à M. Frédéric GAUTHIER-BESNARD, Président
demeurant 454 route de Pernes
84170 MONTEUX
d'une déclaration en date du 24 avril 2006
faisant connaître la constitution d'une association ayant pour titre :
LES GREATS FRAGGEURS

Image originale puis passage en niveau de gris

FIGURE 3.4 – Greyscale

Une fois que l'image a été transformée en niveau de gris, il nous a fallu la passer en noir et blanc. Pour cela, nous avons besoin de définir le seuil de gris à partir duquel nous considérons un pixel comme noir, ou blanc. Nous avons donc effectué quelques recherches pour finalement tomber sur la méthode d'Otsu décrite ci-dessous qui applique une transformation pour chaque pixel en pixel noir ou blanc.

L'avantage de la binarisation est que travailler avec une image binaire est beaucoup plus simple et léger. Cela permet d'identifier les caractères et les différents objets plus facilement.

Par ailleurs, dans nos algorithmes, nous n'utilisons pas de matrices numériques avec des 0 et des 1, mais des matrices booléennes, c'est à dire que le blanc sera représenté par faux et le noir par vrai. Pour l'utilisateur lambda aucune difference, mais pour OCaml, la gestion mémoire d'un tableau à valeurs fixes (du moins contantes, ici vrai ou faux, donc 2 possiblités ce qui peut se représenter en memoire par un seul bit - en fait 8bits en réel) est beaucoup plus aisée que la gestion d'un tableau (dynamique) pouvant contenir n'importe quelle valeur numérique (et donc un nombre de bit beaucoup plus elevé par point de la matrice).

3.3.1 La méthode d'Otsu

La méthode d'Otsu consiste en un seuillage automatique à partir de la forme de l'histogramme de l'image (oui c'est très barbare et incompréhensible). Plus simplement, l'algorithme suppose que l'image à binariser ne contient que deux classes de pixels, (c'est-à-dire le premier plan et l'arrière-plan) puis calcule le seuil optimal qui sépare ces deux classes afin que leur variance intra-classe soit minimale. Pour rappel, la variance est une mesure servant à caractériser la dispersion d'une distribution ou d'un échantillon.

Tout d'abord, des statistiques sont réalisées pour les deux classes de valeurs d'intensité différentes et séparées par un seuil. Ces statistiques sont alors calculées pour tous les seuils possibles et ceci allant jusqu'à i. Une fois ceci fait, la séparation des pixels se fait à l'aide de la moyenne et de la variance dont les formules sont les suivantes :

$$\boldsymbol{histogramme}_{normalisé} = \frac{n_i}{\sum n_i}$$

$$\boldsymbol{moy} = \sum_{i=1}^{k} i \times \boldsymbol{histogramme}_{normalisé}(i)$$

et

$$\boldsymbol{var} = \sum_{i=1}^{k} \boldsymbol{histogramme}_{normalisé}(i)$$

avec n_i représentant le nombre de pixels de niveau i dans l'image.

Pour info, l'histogramme est un graphique allant de 0 à 255 et définissant les niveaux d'une image avec en abscisse la luminosité et en ordonnée le nombre de pixels.

Ensuite, une fois ces valeurs calculées, on applique le calcul suivant pour toutes les valeurs de k :

$$s^2(\boldsymbol{k}) = \boldsymbol{var(k)} \cdot (1 - \boldsymbol{var(k)}) \cdot (\boldsymbol{moy}(255) \cdot \boldsymbol{var}(k) - \boldsymbol{moy}(k))^2$$

$$pour \boldsymbol{k} = 1...255$$

A partir de ce moment-là, la fonction critère sera maximisée par un niveau qui sera considéré comme le seuil pour la binarisation de l'image.

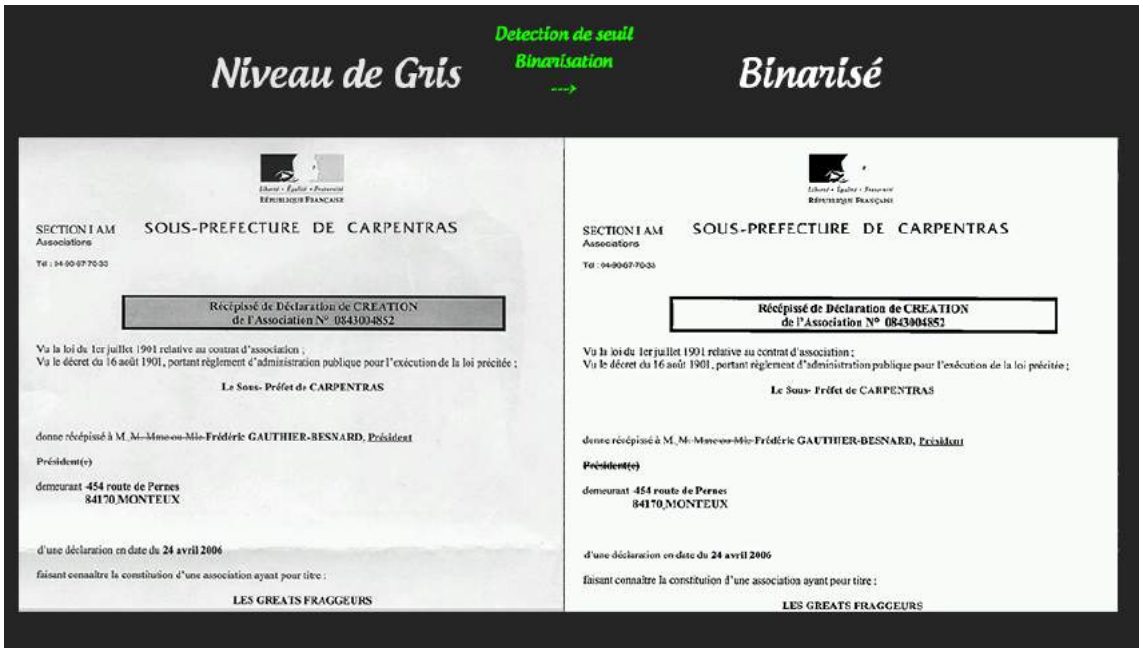

Detection de seuil
Binarisation
--->
Niveau de Gris
Binarisé
SECTION I AM
Associations
SOUS-PREFECTURE DE CARPENTRAS
Récépissé de Déclaration de CREATION
de l'Association N° 0843004852
Vu la loi du 1er juillet 1901 relative au contrat d'association ;
Vu le décret du 16 août 1901, portant règlement d'administration publique pour l'exécution de la loi précitée ;
Le Sous- Préfet de CARPENTRAS
donne récépissé à M. M. Mme ou Mlle Frédéric GAUTHIER-BESNARD, Président
Président(e)
demeurant 454 route de Pernes
84170 MONTEUX
d'une déclaration en date du 24 avril 2006
faisant connaître la constitution d'une association ayant pour titre :
LES GREATS FRAGGEURS

FIGURE 3.5 – Binarisation de l'image

Image avant et après Binarisation

3.4 Filtres

Dans cette partie, nous allons parler des filtres, ce qu'ils sont, dans quoi ils sont utilisés, différents filtres existants, lesquels nous avons choisis et pourquoi nous les avons choisis.

3.4.1 Qu'est ce qu'un filtre ?

Les filtres sont indispensables à tout ce qui touche au domaine du traitement d'images. Ils sont en fait des transformations de l'image d'origine, afin d'arriver à une image plus nette, ou plus floue, des bords plus marqués, etc...

Les filtres que nous utiliserons pour ce projet seront majoritairement fait pour le gommage qui permettra de rendre l'image plus "propre", c'est-à-dire d'enlever des petites taches qui seraient apparues lors de la numérisation du document par exemple.

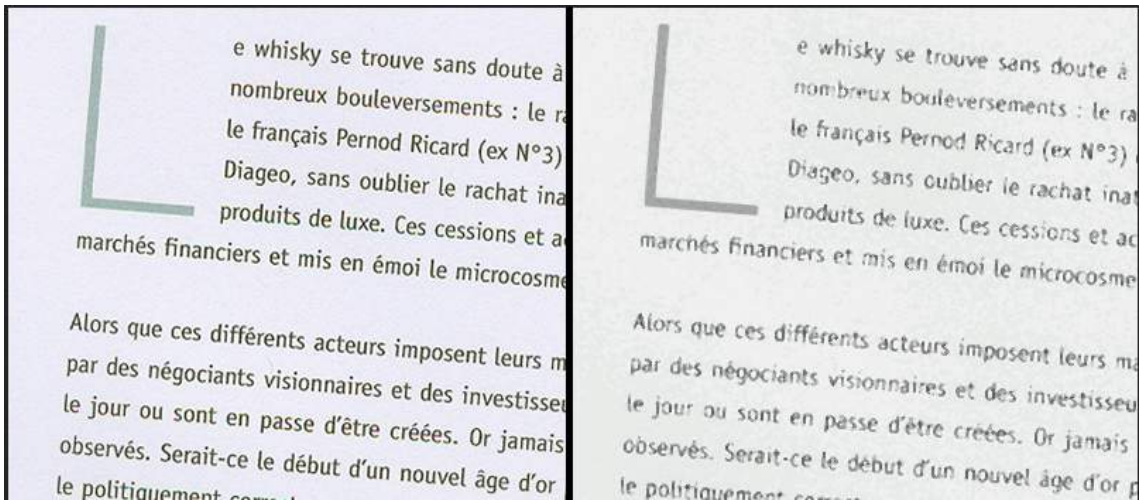
Nous avons utilisé pour l'instant deux types de filtres : le filtre médian et les matrices de convolutions. Nous avons testé les filtres linéaires mais nous avons fini par ne plus les implémenter pour les raisons expliquées dans la section suivante.

3.4.2 Filtre linéaire

Même si nous avons fini par ne plus l'implémenter, nous allons vous dire pourquoi ce choix. Tout d'abord nous allons expliquer ce que le filtre linéaire applique à notre image. Pour chaque pixel de l'image nous prenons les 8 pixels autour de lui et nous les mettons dans une matrice, ensuite nous multiplions celle-ci à une matrice de convolution, c'est-à-dire une matrice de taille n contenant des coefficients. Et enfin nous obtenons la nouvelle valeur du pixel choisi en calculant la moyenne pondérée des valeurs obtenues via le masque de convolution. Le problème de ce filtre est que nous voyons l'apparition de l'étalement des couleurs. En effet, lors du calcul de certains pixels, il se peut que ceux-ci optiennent une nouvelle couleur qui affectera ensuite d'autres pixel et ainsi de suite. Nous obtenons donc à la fin une image avec un gros effet de flou qui rend difficile l'extraction des caractères.

3.4.3 Filtre médian

Le filtre médian est efficace car il arrive à la fois à bien supprimer le bruit impulsionnel tout en préservant assez bien la qualité de l'image. Pour appliquer le filtre, pour chaque pixel, nous prenons les 8 pixels qui l'entourent, ensuite nous mettons tous ces pixels dans une liste, puis nous déterminons la valeur médiane. Nous remplaçons ensuite le pixel central par cette médiane. Ceci est le fonctionnement du filtre médian, toutefois, pour la deuxième soutenance nous avons préféré utiliser une des variantes du filtre médian qui s'appelle le filtre médian relaché car il s'avère que le filtre médian s'avère un peu trop sevère car dans le cas de traits fins, il arrive que le filtre élimine ceux-ci partiellement voir complètement ce qui peut facilement fausser les résultats finaux, par conséquent nous avons choisi le filtre médian relaché car comme son nom peut l'indiquer, celui-ci a une tolérance plus prononcée envers ces détails.


e whisky se trouve sans doute à
nombreux bouleversements : le r
le français Pernod Ricard (ex N°3)
Diageo, sans oublier le rachat ina
produits de luxe. Ces cessions et a
marchés financiers et mis en émoi le microcosme
Alors que ces différents acteurs imposent leurs m
par des négociants visionnaires et des investisse
le jour ou sont en passe d'être créées. Or jamais
observés. Serait-ce le début d'un nouvel âge d'or
le politiquement
e whisky se trouve sans doute à
nombreux bouleversements : le ra
le français Pernod Ricard (ex N°3)
Diageo, sans oublier le rachat inat
produits de luxe. Ces cessions et ac
marchés financiers et mis en émoi le microcosme
Alors que ces différents acteurs imposent leurs ma
par des négociants visionnaires et des investisseu
le jour ou sont en passe d'être créées. Or jamais
observés. Serait-ce le début d'un nouvel âge d'or p

A Gauche l'image originale, à droite l'image passée au filtre médian

FIGURE 3.6 – Filtre Médian

3.4.4 Les Matrices de convolution

Avant de parler de matrice de convolution, il est important tout d'abord de décrire ce qu'est une convolution. Une convolution, dans ce cas, consiste en le traitement d'une matrice par une autre matrice d'entiers de convolution appelée noyau. Ici, nous supposons que qu'une image peut prendre la forme d'une matrice de pixels. A partir de là, nous réalisons un produit de convolution qui revient à multiplier chaque pixel de notre matrice issue de l'image et des 8 pixels qui l'entourent par la valeur correspondante dans le noyau, puis d'ensuite additionner toutes ces valeurs et le résultat sera divisé et le pixel central prend alors la valeur du résultat final.

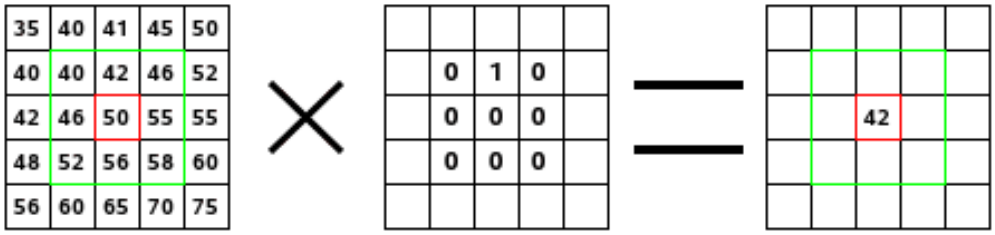


FIGURE 3.7 – Matrices de convolutions

Nous multiplions notre matrice de pixels (à l'extrême gauche) à une matrice de convolution

Dans l'exemple ci-dessus le pixel rouge représente le pixel initialement choisi. Les pixels adjacents à celui-ci sont entourés en vert. La matrice la plus à droite représente le résultat final du calcul qui est 42.

Dans notre cas, l'application diffère de la simple application de fonction sur chaque pixel. En effet, nous avons implémenté les matrices de convolution afin de pouvoir appliquer n'importe quelle matrice à notre image. Là aussi notre cours de mathématiques de SUP nous a été très utiles concernant l'application d'une matrice à une autre. Autre précision, la matrice de convolution que nous allons utiliser est le filtre gaussien qui donne des résultats acceptables car supprimant le bruit moyennement bien mais qui rend l'image assez floutée comme vous pouvez le voir dans l'exemple qui suit.

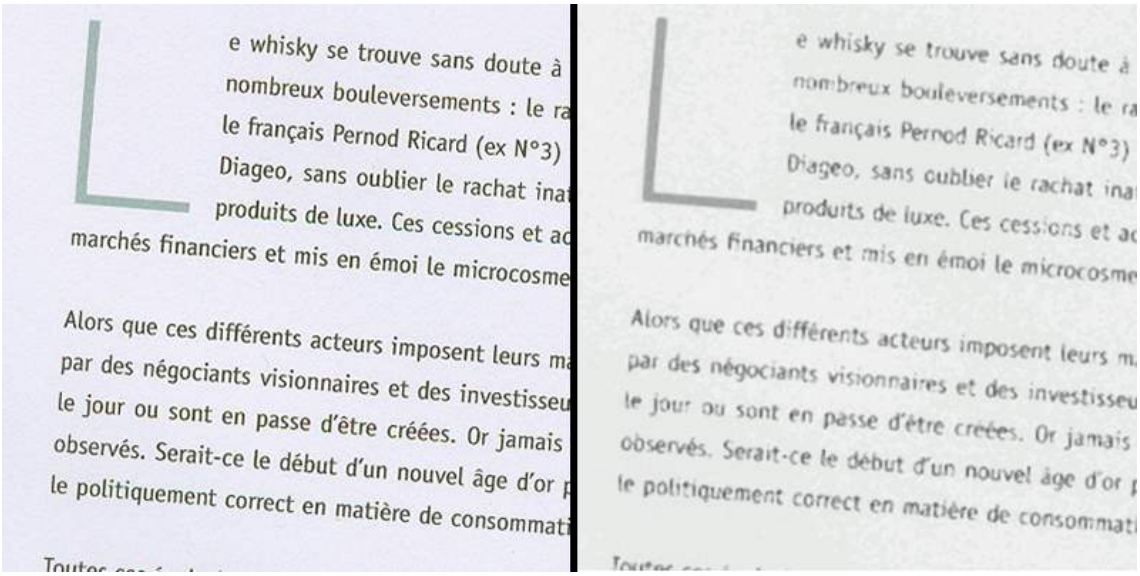

e whisky se trouve sans doute à
nombreux bouleversements : le ra
le français Pernod Ricard (ex N°3)
Diageo, sans oublier le rachat inat
produits de luxe. Ces cessions et ac
marchés financiers et mis en émoi le microcosme
Alors que ces différents acteurs imposent leurs ma
par des négociants visionnaires et des investisseu
le jour ou sont en passe d'être créées. Or jamais
observés. Serait-ce le début d'un nouvel âge d'or
le politiquement correct en matière de consommati
e whisky se trouve sans doute à
nombreux bouleversements : le ra
le français Pernod Ricard (ex N°3)
Diageo, sans oublier le rachat ina
produits de luxe. Ces cessions et ac
marchés financiers et mis en émoi le microcosme
Alors que ces différents acteurs imposent leurs m
par des négociants visionnaires et des investisseu
le jour ou sont en passe d'être créées. Or jamais
observés. Serait-ce le début d'un nouvel âge d'or
le politiquement correct en matière de consommati

FIGURE 3.8 – Filtre Gaussien

A Gauche l'image originale, à droite l'image passée au filtre Gaussien

Chapitre 4

Extraction des caractères

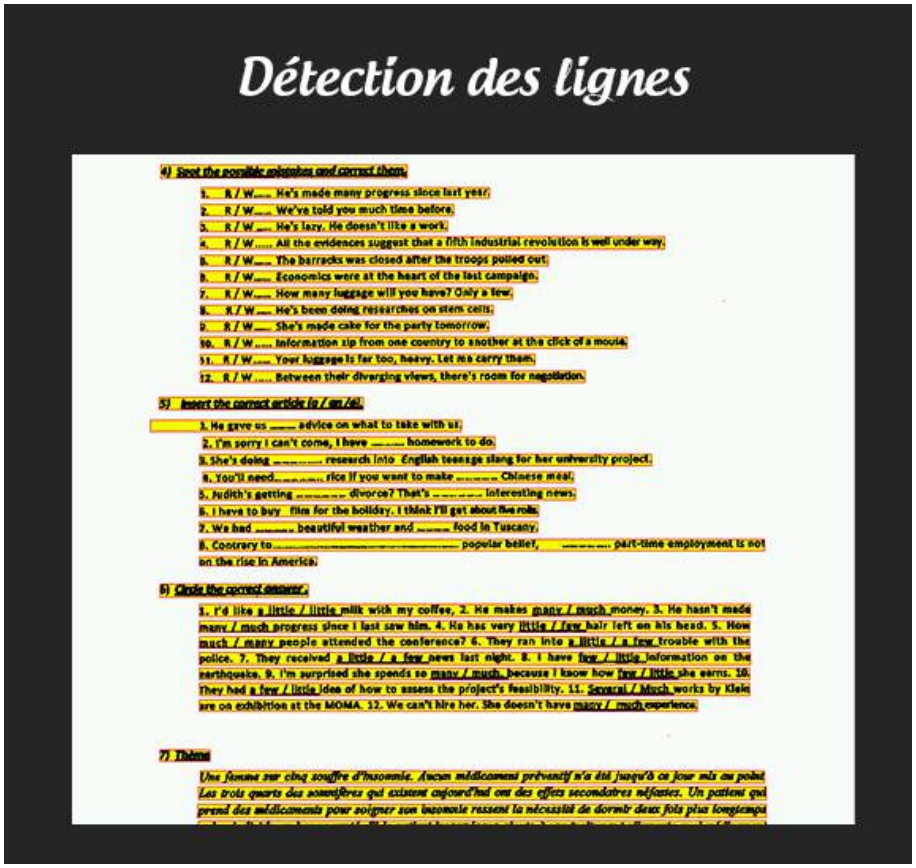
Cette partie comprend la détection des différentes parties du texte, des lignes et des caractères. Les façons dont nous y sommes arrivés et un résumé de nos précédents essais.

4.1 Détection des blocs

Pour détecter les lignes, nous avons dû commencer par la détection des blocs. Pour cela, nous avons utilisé un algorithme assez connu du nom de RLSA (Run Length Smoothing Algorithm). Cet algorithme permet de segmenter une image en région homogène. Elle parcourt une première fois la matrice ligne par ligne puis une seconde fois cette fois-ci colonne par colonne. Lors du parcours, lorsque l'on tombe sur un pixel noir, nous allons poursuivre le parcours en comptant le nombre de pixels blancs qui suivent jusqu'au prochain pixel noir ou le bord de l'image. Si ce nombre a dépassé un seuil prédéfini par nos soins alors nous colorons les pixels blancs en pixels noirs. Une fois tout ceci fait et dans les deux sens, nous obtenons des blocs de couleur noire dont les coordonnées sont facile à récupérer. Dans notre cas, nous appliquons le RLSA a une séquence binaire ou les pixels blancs sont représentés par des 0 et les pixels noirs par des 1.

4.2 Détection des lignes

Dans le domaine de la détection des lignes, il existe beaucoup d'algorithmes comme par exemple l'histogramme de projection ou la transformée de Hough. Lors de notre première soutenance, notre groupe a préféré implanter son propre algorithme afin de détecter les lignes et les caractères. Cet algorithme fait une légère rotation de 1 degré à chaque fois pour un total de 180 fois et vérifie à chaque fois le nombre de blocs sur l'image. L'angle qui obtient le maximum de bloc non seulement est l'angle de rotation mais aussi l'image qui sera traitée.



Exemple de détection des lignes

FIGURE 4.1 – Détection des lignes

4.3 Détection des caractères

Quant à la détection des caractères, nous y avons réfléchis mais nous ne l'avions pas encore implémenté pour cette soutenance-ci. La méthode que nous avons est la suivante : Nous prenons chaque ligne détecte précedemment, puis nous parcourons jusqu'à trouver un pixel noir, depuis ce pixel noir, nous regardons tous les pixels adjacents à celui-ci, grâce à cela, nous pouvons ainsi trouver la position en x et y minimum et la largeur et la hauteur du caractère puis nous passons au suivant jusqu'à parcourir toute la ligne.

Nous avons enfin réussi à avoir une détection des caractères performante pour cette dernière soutenance. Nous avons tout d'abord une fonction qui détecte les blocs comme auparavent, et ensuite nous appliquons une fonction qui permet de séparer ces blocs en lignes dont les coordonnées seront passées dans un tableau avec la hauteur et la largeur de chaque ligne dans celui-ci.

Une fois tout ceci fait, avec les coordonnées que nous avons enregistrées précedemment nous vérifions en premier pour chaque pixel noir de la ligne étudiée les 8 pixels adjacents au pixel. Avant de passer à un autre pixel, nous mettons celui-ci dans une matrice et nous le transformons en pixel blanc. Si ses pixels adjacents sont noirs, nous passons également la fonction sur ceux-ci et ainsi de suite jusqu'à ce que tous les pixels adjacents de chaque pixel soient blancs. Une fois ceci fait, nous obtenons une matrice contenant les pixels noirs déjà traités, puis nous passons au prochain pixel noir non traité de la ligne et nous réitérons la méthode sur toutes les lignes jusqu'à atteindre la dernière ligne.

Chapitre 5

Reconnaissance des caractères

5.1 Réseau de neurones

La reconnaissance de caractères est une étape très importante de notre projet d'Ocr. En effet, à l'aide des images des lettres préalablement découpées dans le document initial, il faut que nous puissions déterminer de quel caractère il s'agit. Cela inclue donc les imperfections qu'il puisse y avoir sur l'image. De plus, il faut qu'il s'adapte également à différentes polices. Pour cela, nous aurons recours à un réseau de neurones. Nous allons donc décrire dans cette partie ce qu'est un réseau de neurones et comment il fonctionne.

5.1.1 Qu'est ce qu'un réseau de neurones ?

Un réseau de neurones est une méthode de calcul, inspiré plus ou moins du fonctionnement des neurones dans notre cerveau. Une de ses caractéristiques principales est qu'il puisse apprendre par l'expérience. Cela veut dire que plus le réseau aura exécuté de fonctions, moins la marge d'erreur sera grande. Les réseaux de neurones biologiques permettent de nombreuses applications comme la reconnaissance de formes, l'apprentissage, la mémorisation etc... Et le réseau de neurones que nous programmons doit donc se rapprocher de celui-ci.

5.1.2 Fonctionnement

Notre réseau de neurones est un algorithme qui prend en paramètre l'image après avoir été traitée préalablement et un tableau contenant pour chacune ses cases un caractère qui a été extrait. Et en sortie, nous obtenons un tableau. L'algorithme lance une fonction qui prend en paramètre une matrice 64*64 représentant 1 caractère avec un poids. Il y a deux parties importantes dans notre réseau de neurones. L'entrainement et la reconnaissance qui peuvent s'apparenter à l'apprentissage et à l'application de ce qui a été appris. Pour la partie entrainement, nous avons un grand tableau avec plein de matrice 64*64 qui s'apparentent à des neurones et chacune de ces matrices étant apparentée à un symbole (a, b, c, d...).

À chacune de ces matrices est associé des poids que l'on peut représenter sous la forme de pixels blancs ou noirs et lorsque nous passons un caractère à une matrice, celui-ci modifiera d'un pas ses poids. Si on devait le représenter par des nombres, on pourrait dire que l'on incrémente de 1 le poids lorsque l'on tombe sur un pixel noir et que l'on décrémente de 1 le poids lorsque l'on tombe sur un pixel blanc. A force de passer un même caractère sous différentes formes, nous arrivons à avoir faire apprendre au neurone à reconnaître car les pixels ayant des poids très forts ou très faibles au final seront ceux que l'on ne peux ignorer lors de la reconnaissance de caractère. Pour la partie reconnaissance, nous prenons tout simplement la matrice ayant eu le maximum de poids en commun avec le caractère concerné.

5.2 Présentation de l'algorithme

L'algorithme de reconnaissance de caractère n'est pas un vrai réseau de neurone, aussi complexe que l'on peut trouver dans les programmes basés sur cette méthode, mais une version plus simple à mettre en place. En effet, nous avons dû chercher des méthodes expliquant ce principe, et avons préféré passer par une première approche non optimisée, mais relativement simple à mettre en place, avant de travailler sur un algorithme plus performant (que nous présenterons à la seconde soutenance)

Le but est ici d'implémenter une fonction qui associe à une image (contenant du texte de préférence...) et à un ensemble de rectangles (les « bounding boxes »des caractères dans l'image) un ensemble de symboles correspondant à l'ensemble des rectangles (ici, de type char).

Comme précisé précedemment, l'algorithme est découpé en deux parties : l'apprentissage et la reconnaissance, qui vont ici être expliquées.

5.3 Donnée centrale

La donnée centrale de l'algorithme est un ensemble de matrice (un tableau de tableaux de tableaux en l'occurence...). Chaque matrice représente un caractère. Elle contient des entiers relatifs, représentant la fréquence d'apparition d'une partie du caractère (et c'est difficile d'être plus clair). Pour être un peu moins abstrait, supposons que chaque caractère ait la même taille, les entiers (appelés poids) de la matrice seront très faibles (valeur négative) si le pixel en question n'apparaît jamais (ou que très peu), et très élevés si ce pixel apparaît fréquemment.

Cet ensemble de matrice doit être stocké en mémoire entre la phase d'apprentissage et la phase de reconnaissance de texte, c'est un simple fichier binaire sur le disque.

5.4 Principe d'apprentissage

Pour s'adapter à n'importe quelle police de caractère, le programme doit « apprendre »différentes variations d'un même caractère. En effet, pour un caractère donné, certains pixels reviendront très souvent, tandis que d'autres que très rarement. Le but va bien sûr de mettre à jour les poids dans la matrice correspondante à un caractère donné. Nous offrons alors deux méthodes d'apprentissage : à partir d'une image ou à partir d'un fichier de police. (pas de la police)

À partir d'une image, le programme va commencer par traiter l'image, et va envoyer à l'algorithme d'apprentissage l'image et la liste des *bounding boxes* de chaque caractère, exactement comme si le texte devait être reconnu. Pour chaque caractère, le programme affiche l'image correspondante, donne le caractère qui lui semble le plus proche, et l'utilisateur doit le corriger si ce n'est pas le caractère

attendu. Les poids dans la matrice sont mis à jour en incrémentant (ou en décrémentant) chaque valeur en fonction de si le pixel fait partie du caractère ou non (en l'occurence, noir ou blanc).

À partir d'une typographie (fichier *.ttf* ou *.odt*), le programme va, pour chaque caractère, effectuer un rendu de ce caractère et ajuster les poids dans la matrice correspondante (suivant le même principe qu'avec une image).

5.5 Principe de reconnaissance de caractère

Reconnaître un caractère n'est pas compliqué : il suffit d'extraire chaque caractère, et d'associer à chaque matrice constituant le « charset »une valeur (un score) calculé très simplement : soit $\phi =$ somme des + ou - poids de la matrice courrante (plus si le pixel de l'image est « activé », moins sinon) et $\mu =$ somme des poids positifs de la matrice. Le quotient $\frac{\phi}{\mu}$ est un nombre compris dans l'intervalle $[0; 1]$. Plus ce quotient est proche de 1, et plus l'image est proche du caractère représenté par la matrice. Oui, c'est aussi simple que cela.

5.6 Inconvénients de cette méthode

Etant relativement simple, ce principe est très loin d'être optimal. En effet, la reconnaissance d'un texte d'une centaine de caractères prend environ 10 secondes. De plus, le fait de stocker tous les caractères dans un fichier sur le disque consomme relativement beaucoup de place (1 Mo maximum, ce qui est considérable par rapport à ce que fait l'algorithme).

Lors de la seconde soutenance, le principe même de l'algorithme a été remis en cause, et remplacé par une méthode basée elle aussi sur un réseau de neurone, mais plus complexe et plus optimisée que celle décrite ici.

5.7 Perceptron

Le perceptron est un réseau orienté de neurones artificiels et pouvant être organisé en plusieurs couches. L'information à travers ce perceptron se déplace de la couche d'entrée ne contenant aucun neurones vers la couche de sortie qui correspond à la sortie du réseau. Les neurones à l'intérieur sont reliés entre eux à l'aide de connexions. Il y a deux types de perceptron, le perceptron monocouche et le perceptron multicouches. Le perceptron mono-couche n'a qu'une seul sortie à laquelle toutes les entrées sont connectées. On appelle ce genre de perceptron feed-forward à l'inverse des perceptrons multicouches que l'on appelle récurrents car ceux-ci alimentent leurs entrées avec leurs sorties contrairement aux premiers.

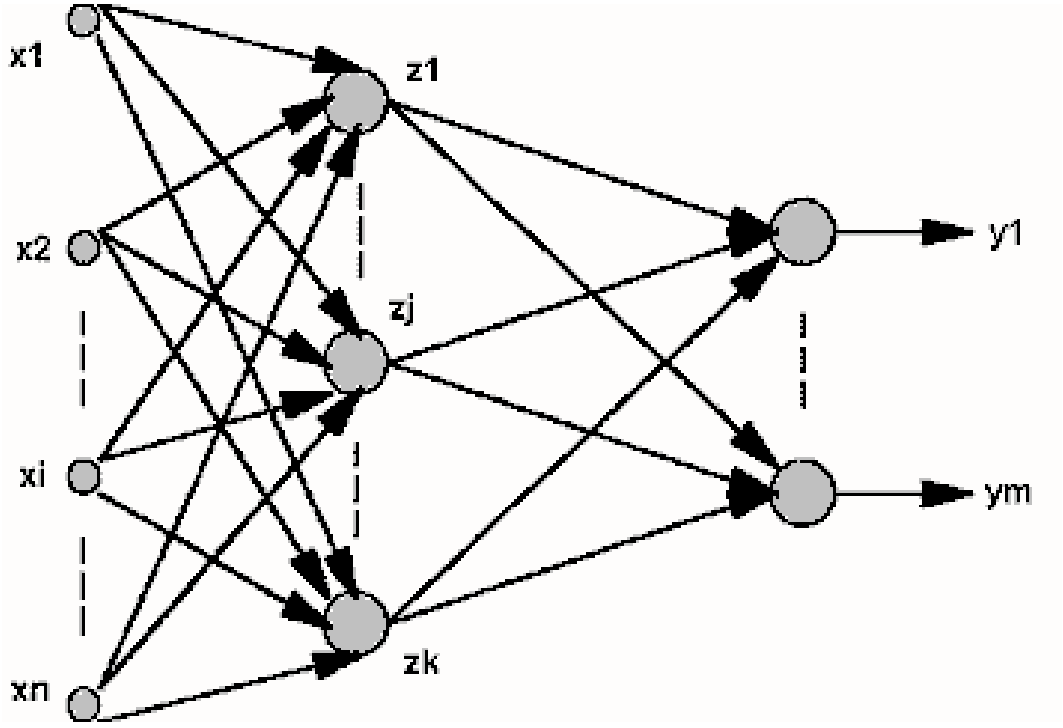


FIGURE 5.1 – Perceptron multicouches

Schéma d'un perceptron multicouches (ou pmc)

Dans le schéma que nous vous présenterons plus bas, nous utilisons un perceptron mono-couche pour chaque caractère. Nous avons opté pour implémenter un perceptron monocouche car malgré le fait qu'un perceptron multicouches soit plus performant, celui-ci est plus difficile à implémenter et pour notre OCR, nous n'avons pas besoin d'avoir un perceptron multicouches.

Ensuite nous appliquons la formule suivante :

$$\boldsymbol{Z} = \sum X_i W_i - \theta$$

Où θ représente le seuil à dépasser pour que la sortie de chaque perceptron soit à 1.

Après avoir fait ceci, nous exécutons une fonction d'activation et qui est était au début la fonction de Heaviside (aussi appelée fonction de seuil) qui prend la forme suivante :

$$\boldsymbol{Y} = H(Z) = \begin{cases} 0 & \text{si } Z < 0 \\ 1 & \text{si } Z \geq 0 \end{cases}$$

Mais ensuite nous avons préféré utiliser la fonction sigmoïde qui est quant à elle définie par :

$$\forall x \in R, \sigma = \frac{1}{1+e^{-x}}$$

Nous avons fini par utiliser la fonction sigmoïde car elle avait la particularité d'être dérivable ce qui s'avère très utile pour elle permet également d'avoir des valeurs intermédiaires (des réels compris entre 0 et 1) contrairement à la fonction de Heaviside qui ne renvoie que 0 ou 1. Toutefois les deux fonctions ont un seuil qui est en $x = 0$ mais qui vaut 1 pour la fonction Heaviside tandis qu'elle vaut $\frac{1}{2}$ pour la fonction sigmoïde.

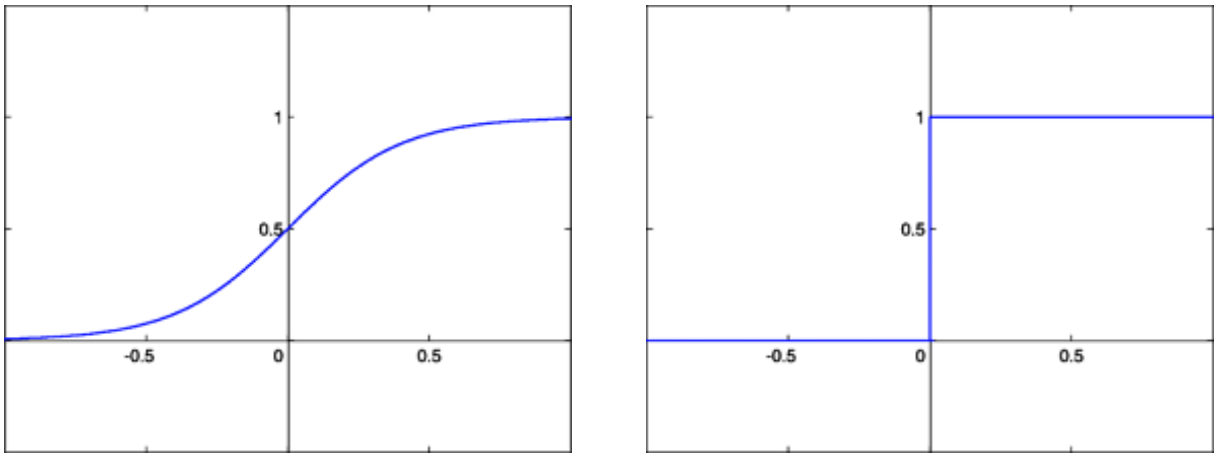


FIGURE 5.2 – Fonction sigmoïde et fonction de seuil

A gauche, la fonction sigmoïde, à droite la fonction de Heaviside.

Malheureusement dans le cas où plusieurs fonctions sigmoïdes renvoient 1, nous devons trouver un moyen de trouver le caractère le plus adéquat. C'est ici que rentre en scène l'algorithme d'apprentissage.

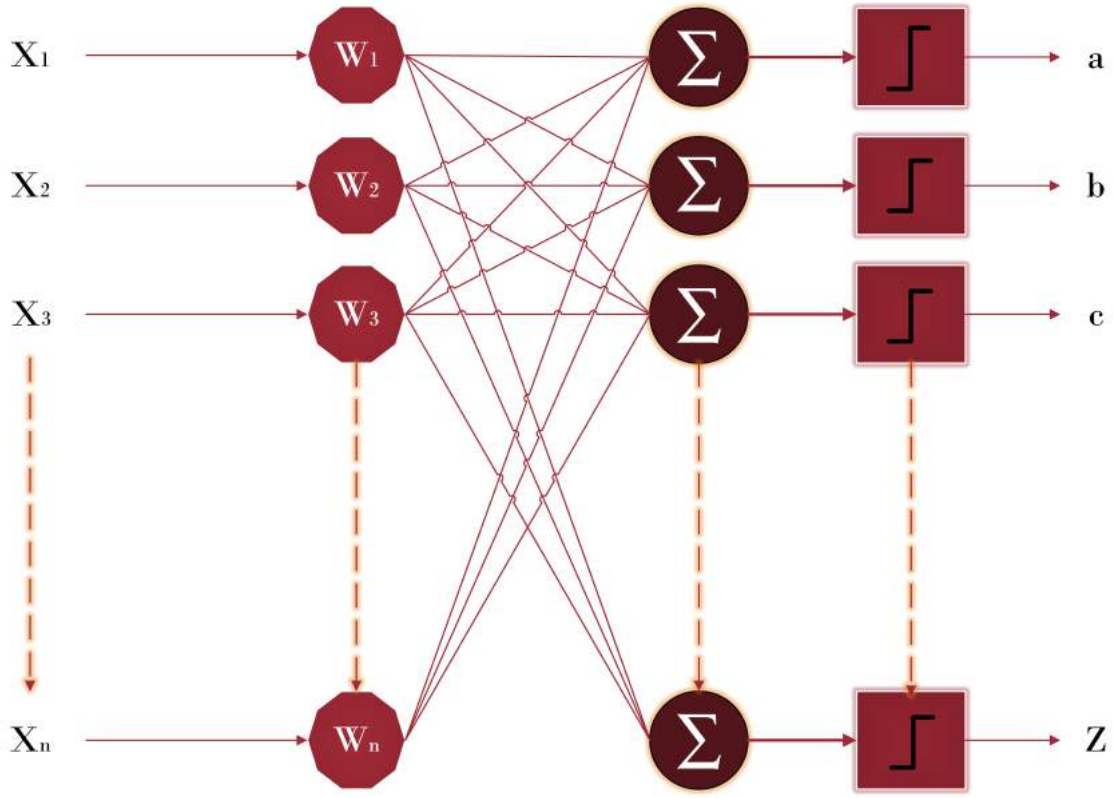


FIGURE 5.3 – Schéma d'un perceptron monocouche

Dans ce schéma, la fonction d'activation est représentée par le signe contenant 2 angles droits.

5.8 Algorithme d'apprentissage

Il existe deux principaux algorithmes pour l'apprentissage à un perceptron monocouche. Le premier et le plus simple par la même occasion se nomme la descente du gradient. Le deuxième, en général plus efficace que le premier, se nomme algorithme de Widrow-Hoff, en référence à ses deux créateurs.

Les deux algorithmes ont pour principe de comparer le résultat obtenu au résultat qui était attendu puis à minimiser l'erreur commise sur les exemples. Toutefois la différence entre les deux sera expliquée un peu plus tard.

L'algorithme de descente du gradient consiste à définir l'erreur quadratique E définit à l'aide de la formule suivante :

$$\boldsymbol{E} = \sum (S_2 - S_1)$$

Avec S_1 les sorties obtenues et S_2 les sorties attendues.

Après avoir obtenu E, nous allons ensuite modifier les poids en faisant en sorte qu'ils se rapprochent des poids "idéaux". Puis nous allons passer plusieurs fois les exemples à chaque neurone. Cette méthode a toutefois un inconvénient, lors de la correction des poids, elle les corrige sur la globalité des exemples, ce qui fait que le temps d'adaptation aux exemples peut sembler assez long.

L'algorithme de Widrow-Hoff, lui, est similaire au premier sauf qu'il a pour différence qu'il modifie les poids après chaque exemple au lieu d'attendre d'avoir tout dans sa globalité ce qui minimisera beaucoup plus l'erreur. Les résultats seront donc par conséquent bien meilleurs et tendront plus rapidement vers le résultat escompté.

Chapitre 6

Interface Graphique

Nous avons commencé à nous préoccuper de l'interface graphique qu'à partir de la deuxième soutenance. Nous avons utilisé Gtk+ pour créer notre interface graphique. Celle-ci a été faîtes en essayant de la rendre la plus facile d'accès possible tout en conservant un maximum de ses possibilités.

6.1 Description des composants

Nous avons une barre horizontale en haut à droite de l'écran contenant des boutons décrivant les diverses étapes de traitement de l'OCR, c'est-à-dire le prétraitement, la rotation, l'extraction des caractères, le réseau de neurones etc... L'écran central de l'interface est séparé en deux. A gauche, nous avons l'endroit ou l'image chargée apparaît et à droite le texte obtenu après traitement de cette image. Nous avons également un bouton permettant d'extraire le texte obtenu sous un fichier .txt. Les résultats des étapes sont affichées clairement sur l'image qui sera modifiée lorsqu'un bouton sera appuyé.

FIGURE 6.1 – Interface graphique

Notre interface graphique après avoir chargé une image

6.2 Utilisation

Après avoir ouvert une image, vous aurez bien evidemment la possibilité d'utiliser notre logiciel d'OCR pour pouvoir extraire le texte contenu dans l'image mais vous aurez également la possibilité de visualiser les différentes étapes qui ont permis à atteindre ce résultat à l'aide des boutons situés dans la barre en haut à droite de l'écran. Il vous sera ensuite possible après avoir utilisé notre correcteur orthographique pour corriger les éventuelles fautes qui seraient apparues lors du traitement d'extraire le texte issu de l'image dans un fichier .txt.

6.3 Gtk+

Tout d'abord, qu'est ce que Gtk+ ? Gtk+ (The Gimp Tool Kit) est une librairie utilisée pour créer des interfaces graphiques. Notre choix s'est porté sur cette librairie pour plusieurs raisons qui sont qu'elle est gratuite, qu'elle est multi-plateforme (même si dans notre cas ce point n'était pas important) et qu'elle est accessible dans plusieurs langages de programmations comme le python, le C et le C++ et bien sûr le Caml.

La création sous Gtk+ consiste à créer et manipuler des objets graphiques de Gtk+. Nous appelons ces objets des "Widgets". Ces Widgets contiennent des propriétés qui permettent la manipulation des divers objets comme par exemple la création de boutons, la création de barre de déplacement ou alors le chargement d'une image. Ils sont donc très utiles et important dans la création de l'interface graphique. Même si nous avons eu du mal au départ car la documentation sur LablGtk (en particulier les tutoriaux) se font rares, nous avons quand même réussi à implémenter l'interface graphique affichée plus haut.

6.4 Correcteur orthographique

Dans le but de corriger les éventuelles erreurs générées lors de la reconnaissance de caractères, nous avons décidé d'implémenter un correcteur orthographique. Le correcteur orthographique que nous utilisons est Gtkspell qui est déjà implémenté par défaut sur Gtk+ ce qui nous a rendu la tache plus simple. Ce correcteur orthographique souligne les mots dont l'orthographe est douteuse et vous donne une liste de mots proches de ceux-ci. Les résultats sont bons et une très grande majorité des mots de la langue française sont présents. Mais avant d'avoir découvert Gtkspell, nous utilisions la distance de Levenshtein.

6.4.1 Principe

Nous disposions d'un dictionnaire de base pour la correction. Celui-ci etait mis dans un fichier texte et avait été utilisé lors de la vérification de l'orthographe d'un mot via la distance de Levenstein.

6.4.2 Distance de Levenshtein

Afin de trouver si un mot est bien orthographié, nous calcule sa distance d'édition. La distance de Levenshtein est la différence entre le mot écrit et le mot le plus proche de lui dans le dictionnaire. Chaque changement de lettre, ou de lettres en trop ou en moins ajoute 1 à la distance. Exemple : "Bouteille" et "Boutaile" ont une distance d'édition de 2. Ici, la distance de Levenshtein nous avait permis de calculer le nombre minimum d'opérations simples à effectuer pour passer d'un mot à un autre. Nous cherchions le mot avec la distance la plus courte du mot donné et nous remplacions le mot erroné par celui-ci. Il y avait toutefois un inconvénient à cela, c'est le moment ou il y avait plusieurs mots étant à même distance du mot. Notre correcteur ne pouvait pas savoir lequel était le mieux adapté dans le contexte de la phrase. Puis nous avons découvert Gtkspell qui fut beaucoup plus simple à implémenter.

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

FIGURE 6.2 – Distance de Levenshtein

Dans l'exemple ci-dessus, nous comparons les mots "elephant" et "relevant". La distance dans ce cas étant de trois, indiquée par le chemin d'erreur en jaune. La zone bleu-gris représente les autres cas possibles mais dans notre cas, nous voulons la distance la plus courte.

Chapitre 7

Problèmes rencontrés

N'ayant pas mis notre code en commun, nous avons eu un certain mal à nous adapter au départ. Mis-à-part cela, le fait que nous ayons à la fois cours et soutenance ensemble nous ont obligé à bien savoir gérer le temps. Ceci ne concerne toutefois qu'avant notre soutenance, après cela, nous avons dû rencontrer d'autres problèmes qui sont que l'execution du code mettait parfois beaucoup de temps et des problèmes lors de l'utilisation de certains modules. De plus nous avions un problème sur nos racks car le code ne compilait pas dessus. Le problème de performance a toutefois été corrigé lorsque nous sommes passés de Ocaml C à Ocaml OPT qui nous a fait gagné un gain de performance considérable (+ de 80%). Du côté de l'interface graphique, nous avons mis un bout de temps avant de réussir à charger une image. De plus, notre fonction de normalisation pour que chaque caractère puisse être réadaptée à la bonne taille n'existe pas. Mis à part tout cela, nous n'avons pas rencontré de gros problèmes en général.

Chapitre 8

Site web

Notre site internet se trouve à l'adresse suivante : http ://dcipher.42portal.com/
Celui-ci est divisé en plusieurs parties qui sont :
-Le projet : Décrit les differentes étapes et les objectifs que nous nous sommes fixés.
-Le groupe : Offre une description brève de chaque membre de notre équipe.
-Téléchargements : Permet de télécharger les différents contenus que ce soit notre projet lui-même ou nos rapports de soutenance.
-Contacts : Toutes les façons de nous contacter.

FIGURE 8.1 – Version 1.0 du site

De plus, pour cette soutenance, nous avons fait un lifting complet de notre site qui est désormais beaucoup plus beau et intuitif. Le logo a été changé. De plus, toutes les parties ont été détaillées et une nouvelles section ont fait leur apparition pour cette seconde soutenance.

Les sections et changements effectués sur celles-ci sont les suivantes :
-Le projet : Remplit la même fonction qu'à la première soutenance.
-News : Cette section donne toutes les actualités concernant notre OCR et notre site.
-Code Breaker : Cette section remplace l'ancienne section groupe, et décrit notre groupe en général, et nos membres plus précisemment et avec des photos.
-Téléchargements : Vous permet désormais de télécharger des logos de notre projet.
-Contact : Encore plus de façons de nous contacter et des méthodes pour augmenter vos chances de nous contacter.

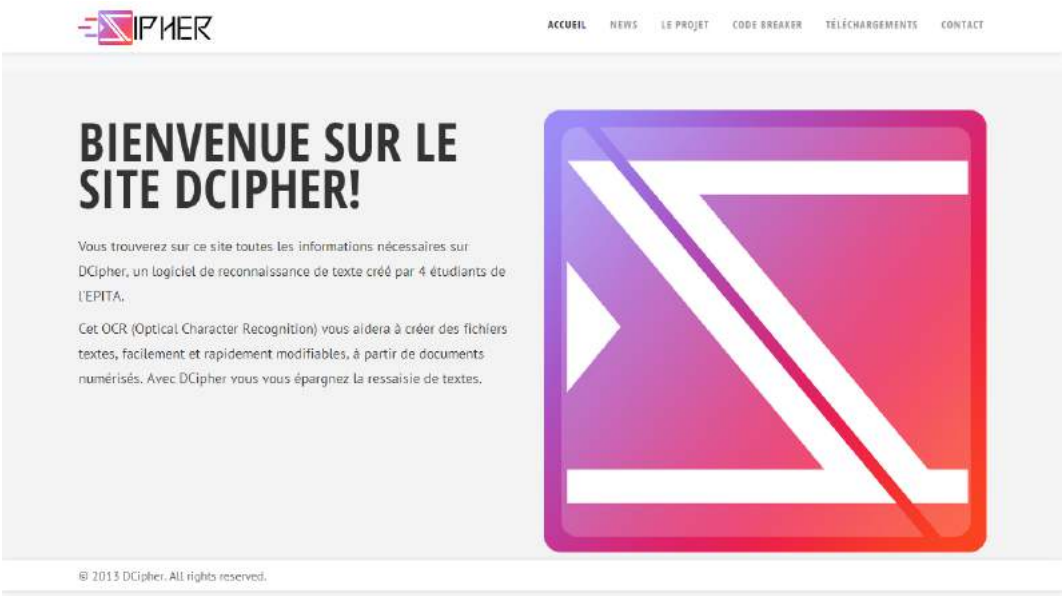

IPHER
ACCUEIL
NEWS
LE PROJET
CODE BREAKER
TÉLÉCHARGEMENTS
CONTACT
BIENVENUE SUR LE SITE DCIPHER!
Vous trouverez sur ce site toutes les informations nécessaires sur DCipher, un logiciel de reconnaissance de texte créé par 4 étudiants de l'EPITA.
Cet OCR (Optical Character Recognition) vous aidera à créer des fichiers textes, facilement et rapidement modifiables, à partir de documents numérisés. Avec DCipher vous vous épargnez la ressaisie de textes.
© 2013 DCipher. All rights reserved.

FIGURE 8.2 – Version 2.0 du site

Version finale de notre site internet

Chapitre 9

Organisation

9.1 Répartition des tâches

Voici la répartition des tâches qui a été faîtes par notre chef de projet en début de projet. Cette répartition est équilibrée entre les différents membres car certaines tâches, comme le réseau de neurones, sont plus complexes que d'autres. De nouvelles tâches ont été attribuées par ailleurs pour notre deuxième soutenance.

X : Doit réaliser cette tâche

Objectifs	Julien	Nils	David	Dat
Greyscale et Binarisation				X
Détection de seuil de binarisation			X	
Filtres			X	
Matrices de convolution	X			
Rotations	X			
Détection des paragraphes et lignes	X			
Reconnaissance des caractères		X		
Réseau de neurones		X		
Interface Graphique				X
Correcteur orthographique			X	
Site web				X

9.2 Avancement de notre projet

Ce tableau est un récapitulatif de notre avancement lors de notre première soutenance.

Objectifs	Etat
Pré-traitement de l'image	Terminé
Détection de l'angle	Terminé
Rotations	Terminé
Détection des paragraphes et lignes	Terminé
Extraction des caractères	Terminé
Reconnaissance des caractères	En avance
Réseau de neurones	Avancé
Interface graphique	Non commencé
Site web	Terminé

Et voici le tableau qui résume notre avancement pour cette soutenance-ci.

Objectifs	Etat
Pré-traitement de l'image	Terminé
Détection de l'angle	Terminé
Rotations	Terminé
Détection des paragraphes et lignes	Terminé
Extraction des caractères	Terminé
Reconnaissance des caractères	Terminé
Réseau de neurones	Terminé
Interface graphique	Terminé
Site web	Terminé

Comme vous pouvez le voir, nous avons réussi à terminer notre projet même si quelques améliorations pourraient se faire ici et là. Mais nous sommes en fin de compte satisfait de ce que nous avons fait.

Chapitre 10

Dossier d'exploitation

10.1 Manuel d'installation

Notre OCR nécessite différents modules,k qui sont les suivants : ocaml sdl, gtk Selon votre distribution, cherchez donc le paquet qui correspond et installez-le avec pkg ou apt-get.

Téléchargez ensuite notre archive à l'adresse suivante :
http ://dcipher.42portal.com/soutenance2.tar.bz2

ou rendez-vous sur notre site en section "Téléchargements". Dézippez l'archive avec "tar -xf soutenance2.tar.bz2" Aller dans le dossier "soutenance2" et faites "make" Si tout s'est bien déroulé un exécutable du nom de "dcipher" devrait être créé. Pour l'utilisation, veuillez suivre le Manuel d'Utilisation.

10.2 Manuel d'utilisation

Pour utiliser notre OCR, rendez-vous dans le dossier de l'exécutable (si vous ne voyez pas de quoi il s'agit, rendez-vous à la section Manuel d'Installation). Par la suite vous pouvez exécuter l'ocr via la console "./dcipher". Les paramètres sont les suivants : –train n (entrainer le reseau de neurones (n iterations)) –imgfile file (reconnaissance de caracteres sur l'image) –fontname font (typo a utiliser pour l'entrainement) –graphical (GUI) –demo str (essaye de lire le texte str convertit sous forme d'image)

Par exemple, téléchargez une police en .ttf, et entraînez l'OCR : "./dcipher –train 100 –fontname police.ttf" Un fichier "ann" contenant les infos de l'entrainement vient d'être créé. Ensuite, lancez l'OCR via l'interface GUI : "./dcipher –imgfile image.jpg –graphical"

Une interface se lance, vous disposez alors de plusieurs options qui correspondent en réalité aux différentes étapes du prétraitement, et de la détection de caractères. Cliquez sur chaque bouton un par un pour obtenir le résultat final.

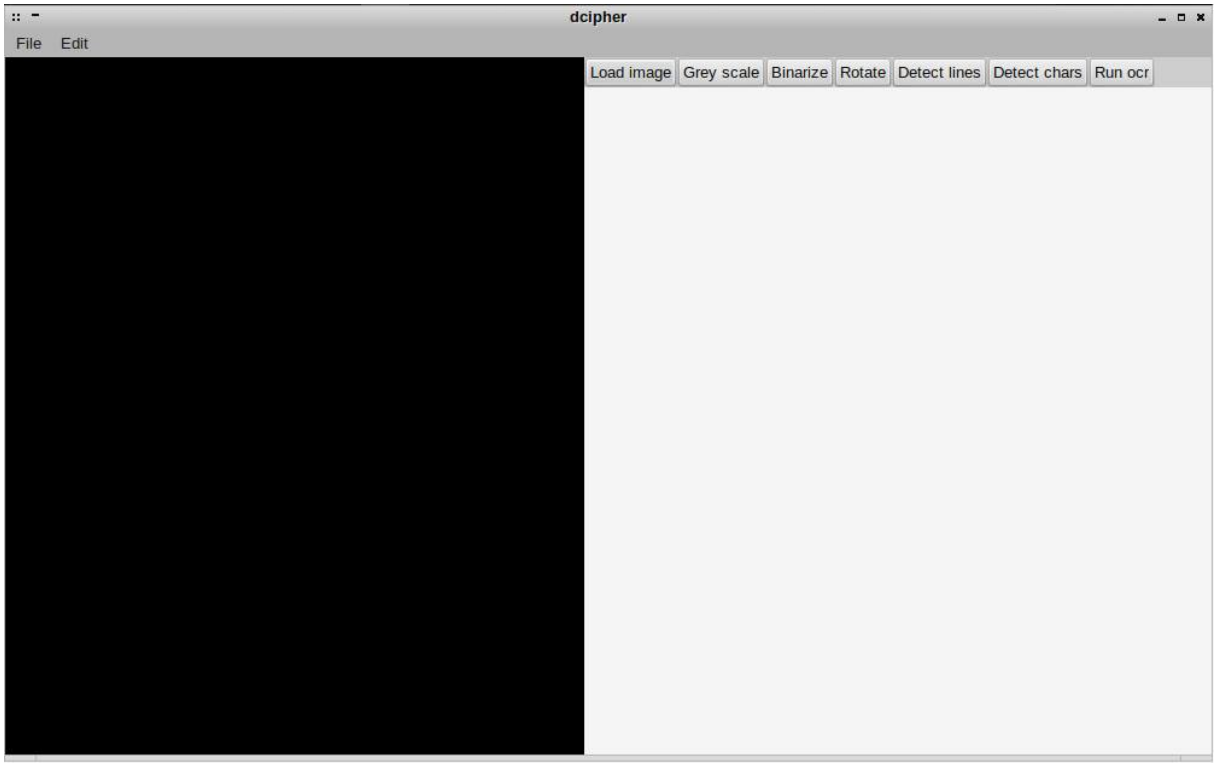


FIGURE 10.1 – Ecran d'accueil de l'interface graphique

Chapitre 11

Impressions personnelles

Cette partie recense les impressions vis-à-vis du projet de chacun de nos membres.

11.0.1 Dubiel Julien

Personnellement, j'ai trouvé ce projet ma foi fort interessant, et les connaissances que j'ai acquises me seront très certainement utiles à l'avenir. Ce projet a été un travail collectif et nous n'y serions sûrement jamais arrivé à temps si nous ne nous étions pas entraidés les uns les autres. Notre emploi du temps était bien plus chargé que l'année dernière et ce projet n'aura pas toujours été une partie de plaisir mais nous y sommes arrivés. Je suis donc heureux de vous présenter notre logiciel OCR fonctionnel, gratuit et facile d'utilisation.

11.0.2 Layet Nils

Ce projet est terminé et je peux dire que je suis satisfait de mon travail. En général, J'ai trouvé ce projet interessant. J'ai pas mal appris lors de ce projet et avoir codé un réseau de neurones a été un bon avant-goût de ce qui m'attend. J'y ai passé beaucoup de temps et j'ai eu des difficultés par moments mais j'ai réussi à force de travail et de perseverance à les surmonter. Mes connaissances dans d'autres langages de programmation m'ont bien aidé et cela m'a également permis d'aider mes camarades de temps en temps.

11.0.3 Florentin Dat

Bon, bah voilà : Enfin terminé ! On peut dire que je me suis donné a fond. J'ai éprouvé beaucoup de difficultés mais grâce à l'aide de mes camarades, j'ai réussi à les surmonter. J'ai appris énormément tout au long de ce projet et je trouve que je me suis bien amélioré et que toutes ces connaissances m'aideront sûrement pour le projet suivant au second semestre ou au futurs projets pour l'année prochaine. Aller, il est temps pour moi de passer à la prochaine étape.

11.0.4 Jiang David

Ce projet a été une bonne expérience pour moi. J'ai trouvé ce projet beaucoup plus interessant que celui de l'année dernière mais ce n'est qu'un avis personnel. Même si au départ je trouvais que ce n'était pas interessant, au fil du temps, mon intéret pour ce projet n'a fait que grandir. Je suis content qu'on l'ait terminé et que tout marche bien.

Chapitre 12

Conclusion

Et nous voici enfin arrivé à la fin de ce rapport. Nous sommes plutôt satisfaits de notre travail car nous avons réussi à avoir un OCR fonctionnel avec une interface graphique en GTK et remplissant le travail qui lui était demandé. Les délais ont été respectés que ce soit lors de la première ou de cette soutenance-ci. Le site web est régulièrement mis à jour et les liens vers le téléchargement de notre OCR ou de nos rapports sont également disponibles sur celui-ci. Ce projet aura été une expérience mémorable pour nous, cela nous aura permis d'approfondir nos connaissances tout en ayant un réel avant-goût de ce qui nous attend l'année prochaine.

Project DCipher

Chapitre 13

Annexes

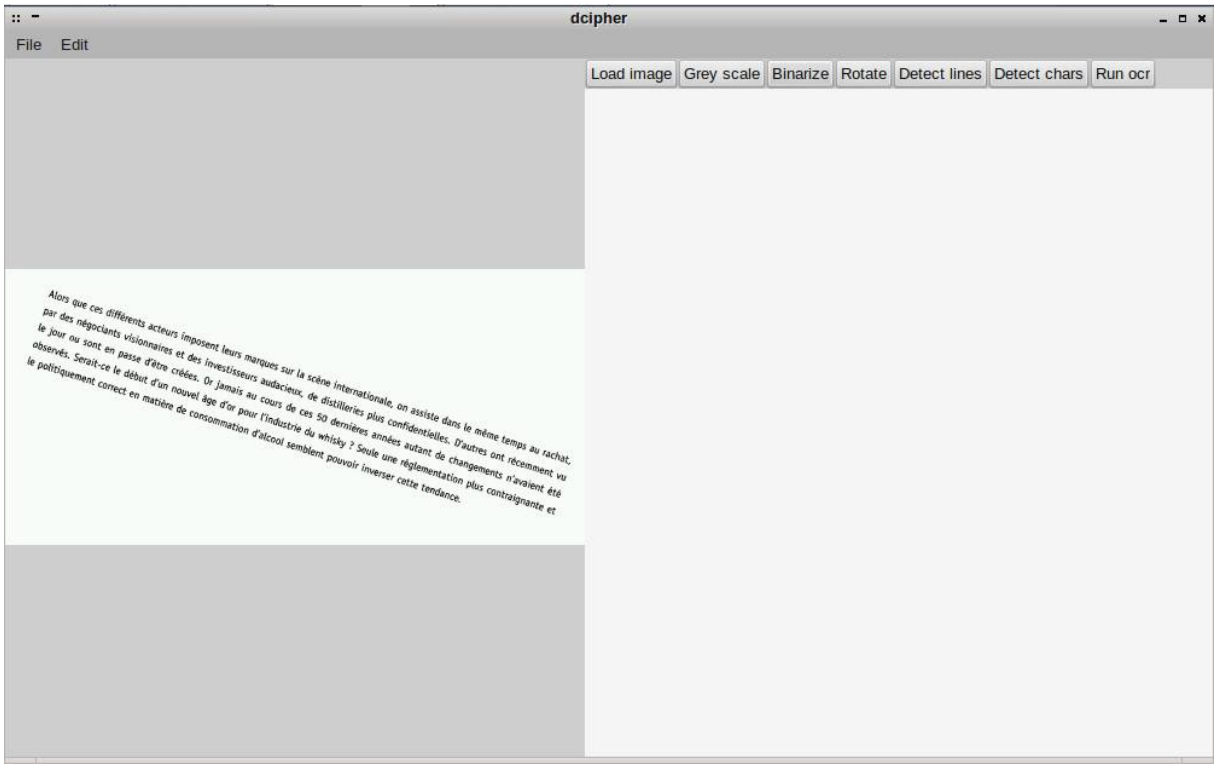

dcipher
File Edit
Load image
Grey scale
Binarize
Rotate
Detect lines
Detect chars
Run ocr
Alors que ces différents acteurs imposent leurs marques sur la scène internationale, on assiste dans le même temps au rachat,
par des négociants visionnaires et des investisseurs audacieux, de distilleries plus confidentielles. D'autres ont récemment vu
le jour ou sont en passe d'être créées. Or jamais au cours de ces 50 dernières années autant de changements n'avaient été
observés. Serait-ce le début d'un nouvel âge d'or pour l'industrie du whisky ? Seule une réglementation plus contraignante et
le politiquement correct en matière de consommation d'alcool semblent pouvoir inverser cette tendance.

FIGURE 13.1 – Interface graphique après avoir exécuté la binarisation

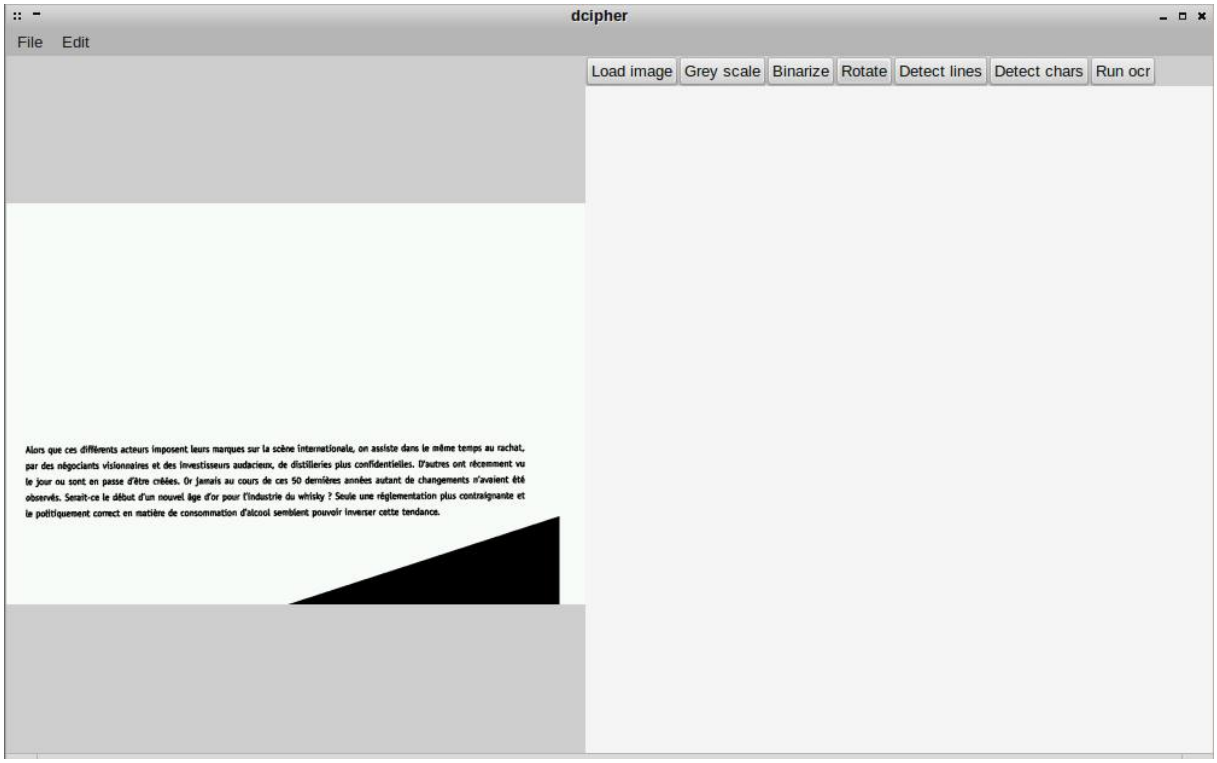

dcipher
File Edit
Load image
Grey scale
Binarize
Rotate
Detect lines
Detect chars
Run ocr
Alors que ces différents acteurs imposent leurs marques sur la scène internationale, on assiste dans le même temps au rachat, par des négociants visionnaires et des investisseurs audacieux, de distilleries plus confidentielles. D'autres ont récemment vu le jour ou sont en passe d'être créées. Or jamais au cours de ces 50 dernières années autant de changements n'avaient été observés. Serait-ce le début d'un nouvel âge d'or pour l'industrie du whisky ? Seule une réglementation plus contraignante et le politiquement correct en matière de consommation d'alcool semblent pouvoir inverser cette tendance.

FIGURE 13.2 – Après utilisation de détection des caractères

FIGURE 13.3 – Ancien logo de notre projet

FIGURE 13.4 – Ancienne bannière de notre projet

FIGURE 13.5 – Nouveau logo de notre projet